Journal of Electronic Testing (2023) 39:571–582
https://doi.org/10.1007/s10836-023-06092-5

Performance Efficient and Fault Tolerant Approximate Adder

Asma Iqbal[1] · **Syed Affan Daimi**[2] · **K. Manjunatha Chari**[3]

Received: 20 June 2023 / Accepted: 31 October 2023 / Published online: 11 December 2023

Abstract
Fault tolerant adders are an important design paradigm to improve the robustness of the adder while at the same time improving the yield. The major downside of fault tolerant adders are the additional modules that are intrinsic to this design. On the other hand, approximate adders take the advantage of computing resilience and inherently improve the area, delay & power metrics. A combination of these two seemingly contradictory approaches are juxtaposed to put forth a design for robust fault tolerant approximate adders that mitigate the effects of redundancy and would help improve the yield. The fault tolerant schemes included are the Triple Modular Redundancy and Partial Triple Modular Redundancy. These are used in conjunction with the approximate Lower part-OR Adder (LOA). The designed fault tolerant approximate adder along with the fault intolerant precise and fault intolerant imprecise adder is used for image sharpening using the Gaussian filter. The results analyzed in the presence and absence of faults indicate that the visual quality of the image in the presence of a single stuck-at fault is almost as good as that obtained without a fault and maintains a PSNR of above 27 in case of fault tolerant approximate adder. There is a significant loss in the image quality if a fault occurs in a non-redundant precise or approximate adder. The deterioration in image quality is more significant if a stuck-at-one fault occurs, as the image becomes visually indecipherable.

Keywords Approximate adders · Fault tolerance · Triple modular redundancy · Partial triple modular redundancy · Image sharpening

1 Introduction

Parallel Adders are a key unit in all processing units and the delay in the adder often determines the delay involved in computations. The Ripple Carry Adder (RCA) is the basic form of a parallel adder. Its major drawback is the delay involved in completing the addition operation due to the movement or rippling of its carry signal from the LSB position towards the MSB. A multitude of designs have been proposed like the Prefix adders, Carry Save adders, Carry Select adders etc. to reduce the delay involved in the RCA and thereby improve the speed of computation.

A new paradigm to improve the efficiency of the adders by compromising on the accuracy of the result to an acceptable level has given rise to the design of approximate adders. This is based on the notion that the effect of MSBs when compared to the LSBs is more significant and hence preserving the MSBs precisely while allowing a tolerable amount of inaccuracy in LSBs would be acceptable. The idea here is to compute the MSBs accurately while permitting a leeway in the computation of LSBs. Approximate computing is found to be extremely helpful in applications like image processing, data mining, multimedia processing, and machine learning which have intensive computations and are resilient to these kinds of errors. This loss of accuracy brings in sizeable improvements in the area, power and specially delay metrics.

Responsible Editor: S. T. Chakradhar

✉ Asma Iqbal
asmadaimi@gmail.com

Syed Affan Daimi
saffand03@gmail.com

K. Manjunatha Chari
mkamsali@gitam.edu

[1] Dept. of Electronics & Communication Engineering, Deccan College of Engineering & Technology, Hyderabad, Telangana, India

[2] Dept. of Computer Science Engineering, Indian Institute of Technology Madras, Chennai, Tamil Nadu, India

[3] Dept. of Electrical, Electronics & Communication Engineering, GITAM University, Hyderabad, Telangana, India

Robustness of the unit computing MSBs is important especially in critical operations [25] like self-driving vehicles and robotics. Hardware redundancy-based fault tolerance is explored in Deep Learning accelerator [14] for critical operations and results indicate an improved reliability. Selective fault tolerance in the majority gates of Quantum-dot Cellular Automata (QCA) approximate adders [22] shows a significant reduction in the average normalized mean error distance. These and similar such works indicate the relevance of reliable approximate computing and to contribute in this domain is the aim of this work.

This paper presents a simple design that combines the advantages inherent in approximate adders and the robustness possible with fault tolerant designs to obtain an adder unit that has highly optimized metrics along with a robust design capable of operating in the presence of faults. This ensures that the MSBs are computed precisely even in case a fault occurs. The designed fault tolerant adder is then utilized to implement an image sharpening application to indicate its efficacy. The stuck-at fault model is used as it models majority of the faults that may occur due to timing constraints and continuous miniaturization.

The arrangement of the paper is as follows. Section 2 gives a brief background on approximate adders and fault tolerant adders with the next section presenting the proposed design along with its simulation results. Section 4 contains the details of the image sharpening algorithm used, the procedural set–up and the inputs used. Section 5 shows the results obtained in the absence and presence of faults for both the non-redundant and the fault tolerant precise and approximate adders with conclusion and future scope in Section 6.

2 Background

As digitization gains greater traction with every passing year, the efficiency of the computation unit which is exceedingly dependent on the adder unit becomes consequential. The improvement in efficiency of adders is being pursued on two fronts. Firstly, efforts are on to design circuits that improve the area, delay and power metrics. Secondly, it is endeavored to improve the reliability and robustness of the unit. These two ideas were succinctly presented in Best-effort computing [3] which put forth three possibilities for computations which were unsuitable for best-effort execution. These are (i) separate hardware and software that provide the traditional "guaranteed" model of computation, (ii) a software overlay layer within the platform that ensures their correct and complete execution on the best-effort substrate, or (iii) by implementing faster, application-specific strategies to compensate for the imperfections in the computing platform.

In the current work fault tolerance is included that has separate hardware that ensures the traditional "guaranteed" output and approximate computing to achieve faster, application-specific approach for error tolerant applications.

Approximate adders are a class of adders proposed to improve the metrics while compromising on the accuracy of the adder.

2.1 Approximate Adders

In this section a few approximate adder designs are discussed. Approximation is achieved at different levels of abstraction by reducing the logic complexity of the full adder unit. The scope of this work is limited to an n-bit Ripple Carry Adders (RCA) with reduction in the gate level complexity are selected. Approximate adders of 'n' bits are usually divided into two parts – the precise adder comprising of 'n-k' bits for the higher order bits and a k-bit imprecise adder for the'k' lower order bits [32].

1. Approximate Mirror Adders (AMAs): A mirror adder is a simple and useful adder design. Logic reduction is employed at the transistor level to reduce the power dissipation and circuit complexity which in turn give better area and delay metrics. Reduction in the logic complexity at the transistor level based on approximation of the output sum and carry has resulted in five approximate mirror adders (AMA) [6].
2. Approximate XOR/XNOR adders (AXAs): These are based on the 10 T adder using multiplexers implemented with pass transistors and XOR/XNOR gates [13]. It is optimized to a design using 8 T and XNOR gates. This optimization leads to three different designs with different approximations for the sum and carry out signals.
3. Lower –part-OR-adder (LOA) [1]: In this design the sum is approximated by an OR gate (instead of the conventional XOR gate) in the imprecise lower part of the adder. The carry input of the precise adder is connected to the carry in signal of the adder (Cin = 0) or it is generated by an AND operation of the most significant bits of the imprecise part of the adder [15].

All these designs incorporate a loss in accuracy with a significant improvement in design metrics. In some of these designs there is a reduced noise margin making it more sensitive. The approximate adders address one part of the efficiency improvement process. The other aspect that helps improve the efficiency, reliability & robustness of the design, is its ability to generate appropriate outputs in case faults occur. Fault tolerance is thus imperative to improve the overall robustness of the system.

2.2 Fault Tolerance

Fault tolerance in adders has widely been studied and innumerable designs proposed for the same. This design methodology has gained higher prominence as the increasing density of fabrication has resulted in falling yield. Hardware redundancy, time redundancy and a combination of these two have been employed to implement fault tolerant adders. As the name suggests hardware redundancy involves the inclusion of additional hardware to mask/overcome the effect of faults [19, 23, 28]. Time redundancy on the other hand uses the same module more than once. The operands are applied in self-dual form [21] or in shifted form [18] and then the outputs are compared. A combination of the two includes certain additional hardware and a multi-cycle operation in case a fault is detected [5]. Further, different parallel adders like the RCA, Carry Save adder [10] and Parallel prefix adders [7] are the focus of different fault tolerant adder design implementations.

The RCA is a parallel adder that has the simplest implementation but at the same time the major disadvantage of the ripple effect of its carry signal from the lower order bits to the higher order bits. This is also the reason for the operation of the RCA being slow and deteriorating as the word size increases. This drawback is inherently overcome by the use of approximate adders as the imprecise part of the adder is designed to dispense with the rippling effect of the carry signal in it. The simplicity of the RCA and its wide usage in the design of approximate adders is an motivation to implement fault tolerant approximate adders based on it. The fault tolerant schemes used in conjunction with the RCA so as to improve its performance are the TMR-RCA [28] and PTMR-RCA [17]. The TMR-RCA is the classic method used for mitigation of faults in adders and is hence included. The TMR-RCA scheme incorporates fault tolerance in the entire adder, both the precise and imprecise part of the adder. As fault tolerance in the imprecise adder is not really significant especially in the case of approximate computing, the PTMR-RCA scheme is also included so as to implement an efficient adder that is also reliable.

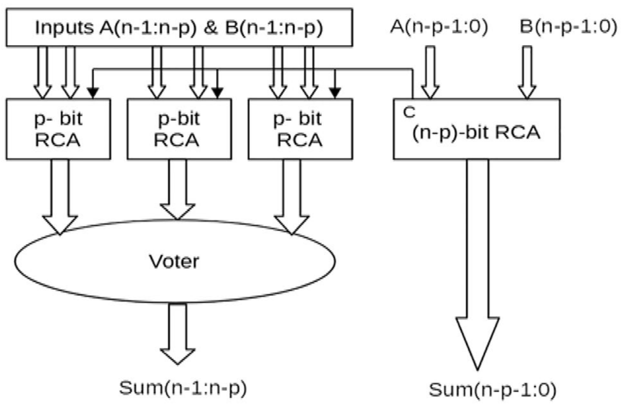


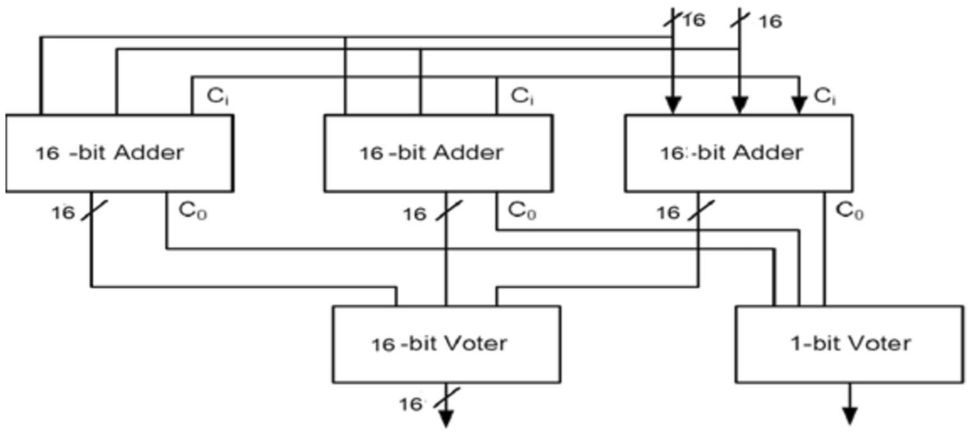
Fig. 2 Block diagram for PTMR

The block diagram of the TMR method to achieve fault tolerance in a Ripple Carry Adder, 16-bit wide is shown in Fig. 1. This requires the triplication of the adder module with each of the module independently generating the output, Sum. The independently generated outputs are then voted upon by the majority voter that gives the final Sum.

This redundancy scheme is modified to obtain the PTMR method of fault tolerance in Ripple Carry Adders. The higher order bits contribute to a larger extent to the sum compared to the lower order bits is one part of the premise on which this is based, the other being that an increase in the hardware makes the entire system more susceptible to faults. These aspects lead to a fault tolerant n-bit adder design in which "p" most significant bits are triplicated while the remaining "n-p" bits are in the simplex mode. There is a considerable reduction in the hardware overhead when compared to TMR. The arrangement of PTMR is given in Fig. 2.

The performance metrics for PTMR-RCA from ten million Monte Carlo simulations [20] measuring Signal to Error Ratio (SER) shows that for low fault rates, $P = 5$

Fig. 1 Block diagram of 16-bit TMR



gives a better performance than the TMR-RCA for a word-size of n = 16. The results also indicate that the advantage of higher SER in TMR reduces as the fault rate increases. The SER is defined as the fraction obtained from the average of the obtained signal power to the variance of its absolute value of error.

SER is given as:

$$\mathrm{SER} = 10\log\frac{\sum_{j=1}^{N}|S(j)|^2}{\sum_{j=1}^{N}\left|E(j)-\widetilde{E}\right|^2}$$

where S is the correct Sum, and S′ the incorrect Sum. E is the magnitude of error E = |S − S′|. N denotes the number of Monte Carlo simulations done. $\widetilde{E}$ is the mean of the error magnitude and E and SER are measured in decibels. A higher value of SER indicates a more robust design.

The observed SER values for varying fault rates are replicated in Table 1. From this it can be inferred that the number of bits that are triplicated i.e., $p = 5$ in PTMR results in an operation similar to TMR in terms of SER.

The value of p can further be reduced in case the fault rates are higher. The detailed comparison of fault rates with SER is included in Table 1 and the results indicate that the PTMR gives a better SER than TMR for $p = 5$ even when the fault rate is low (0.001%). This is related to the hardware reduction in PTMR due to reduced number of bits being triplicated and consequently lesser hardware for the voter circuit. The fault rate which is the rate at which gates fail is considered. For instance, a failure rate of 0.005% translates to 50 failures per million. For higher fault rates the number of significant bits that need to be triplicated can further be reduced.

Table 1 SER of RCA, TMR-RCA & PTMR-RCA vs Fault rates of combinational logic gates (word size = 16 bits) [17]

	Fault %	RCA	TMR	PTMR	
Decreasing fault rate ↓	5.00	9.39	9.86	10.43 (P=2)	Increasing number of triplicated MSBs ↓
	4.00	9.72	10.43	11.15(P=2)	
	3.00	10.29	11.78	12.52 (P=2)	
	2.00	11.64	13.83	14.42 (P=2)	
	1.00	13.98	17.65	18.24 (P=3)	
	0.50	16.52	21.94	22.23(P=4)	
	0.10	23.00	31.19	31.32(P=5)	
	0.05	25.59	34.73	36.80 (P=5)	
	0.01	32.83	42.24	42.27 (P=5)	
	0.001	42.70	52.89	52.60 (P=5)	

3 High Performance Robust Adders

As stated, earlier methods for fault tolerance are included to enhance reliability of the adders. These are used in conjunction with the LOA approximate adder to realize a fault tolerant approximate adder. The initial idea for this design of reliable approximate adders is included in [9].

3.1 Lower Part-OR Approximate Adder

This n-bit adder is divided into two parts- the precise adder comprising of "n-p" bits and an imprecise adder comprising of p-bits. The precise part of the adder consists of "n-p" full adders for the "n-p" bits while the imprecise adder comprises of "p" OR gates for the p-bits. The basic structure is shown in Fig. 3 where A and B represent the addend and the augend. The final output is represented as $\mathrm{Sum}_n\ldots\mathrm{Sum}_0$. As the inaccurate sub-adder is designed using simple OR gates the rippling of the carry is eliminated while at the same time reducing the hardware requirements from this part of the adder. This is the major contributor to the inherent improvement in speed, area and power requirements in an LOA approximate adder.

3.2 Quantitative Analysis

A number of Full adder structures have been implemented and a few are included in this quantitative analysis where we compare the different implementations and their optimizations when used in realizing approximate adders.

The CMOS adder architectures compared are the 10T [4], 14T [27], CPL [30], TFA [2], TGA [24], C-CMOS [29], HYBRID [31] and FA24T [26]. The 10T CMOS adder requires lesser silicon area but has low driving capability. This issue is resolved by using the 14T adder. The Complementary Pass-Transistor (CPL) adder requires 32 transistors and has good voltage swing restoration. A 16 transistor Transmission Function Full adder (TFA) is realized using the transmission function theory. The Transmission Gate

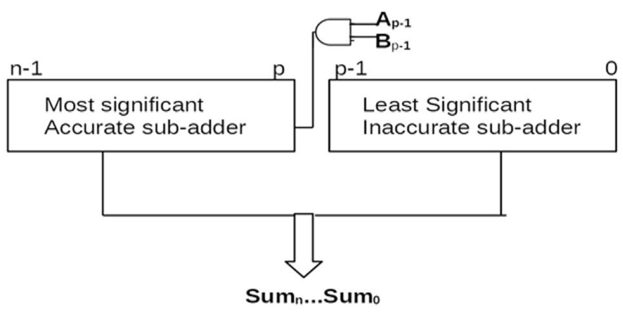

A_{p-1}
B_{p-1}
n-1
p
p-1
0
Most significant Accurate sub-adder
Least Significant Inaccurate sub-adder
Sum_n...Sum_0

Fig. 3 LOA approximate adder

Table 2 Power, delay and PDP for adder structures [8]

Adder structure	Transistor count	Power (μW)	Delay (pS)	PDP (aJ)
10 T	10	0.99	101.51	100.49
14 T	14	1.72	105.86	182.08
CPL	32	2.79	54.21	151.25
TFA	16	0.97	88.20	85.55
TGA	20	1.01	83.29	84.12
C-CMOS	28	1.31	66.28	86.83
HSPC	22	1.42	58.82	83.52
FA24T	24	1.18	82.65	97.53

Adder (TGA) contains 20 transistors and is based on a logic similar to pass transistor logic. However, it successfully overcomes the voltage degradation in CPL adders. It comprises an NMOS and a PMOS controlled by complementary signals. The Complementary CMOS (C-CMOS) adder contains 28 transistors and its reliability while transistor sizing and voltage scaling are its unique advantages. The mirror adder has the same transistor count as the C-CMOS but it has a reduced carry propagation path delay. The Hybrid pass logic with CMOS output drive full adder (HSPC), in which XOR and XNOR functions are generated at the same time using pass transistor logic produce full-swing output at the cost of increased delay. The comparison of the power, delay and area metrics [8] using 45nm CMOS technology are replicated in Table 2. The different adders included in the table vary in the area (based on transistor count), speed of operation and power requirements. The selection of a particular adder structure is done evaluating the requirements of an application in conjunction with the advantages and disadvantages of a particular adder design.

The quantitative analysis of these adders when used in implementation of precise and approximate adders on the basis of transistor count is carried out. This gives dependable results as standard digital blocks are used in the implementation of the design.

The different adder architectures can be compared on the basis of the number of transistors or gates required for implementation or on the basis of technology. Since the adder architectures are usually designed and compared based on the transistor count, this method is utilized. It is used to compare these adder architectures to appreciate the reduction in hardware in case an approximate adder is used instead of a precise adder. The OR gate required for implementing the imprecise adder is implemented using the CMOS logic of NOR followed by a NOT gate. The adders with that are quantitatively compare are of 16-bit and 32-bit word-size. The width of the precise adder in case of a 16-bit adder is taken to be n/2 i.e., 8-bits and the imprecise adder of n/2 bits i.e., again 8-bits. For the 32-bit adder two comparisons are carried out. In the first case the width of the precise and imprecise sub-adder units is taken to be n/2, i.e., each sub-adder is 16-bit wide. In the second case the precise sub-adder is taken to be 8-bit wide with the imprecise sub-adder being 24-bit wide. For actual applications the division of the adder into the precise and imprecise sub-adders is based on requirements such as accuracy speed and power. The comparisons based on transistor count are presented in Table 3. The reduction in hardware is computed as:

$$\frac{\text{Transistor count }[\text{precise adder} - \text{approximate adder}]}{\text{Transistor count }[\text{precise adder}]}$$

The LOA Approximate adder is augmented with fault tolerance to achieve the second objective in the design of high-performance robust adders.

3.3 Fault Tolerant Approximate Adders

In this section two fault tolerant approximate adder designs are presented. The first design achieves fault tolerance

Table 3 Hardware requirement comparisons of precise adder vs approximate adder

Adder structure	Transistor count		Total transistor count					% Hardware reduction		
	Adder	*OR*	*Precise adder*		*Approximate adder*			*k = 8 bits*		*k = 16*
			16-bit	*32-bit*	*16-bit (k = 8)*	*32-bit (k = 8)*	*32-bit (k = 16)*	*16-bit*	*32-bit*	*32-bit*
10 T	10	06	160	320	128	256	224	20.0	20.0	30.0
14 T	14	06	224	448	160	320	256	28.6	28.6	42.9
CPL	32	06	512	1024	304	608	400	40.6	40.6	60.9
TFA	16	06	256	512	176	352	272	31.3	31.3	46.9
TGA	20	06	320	640	208	416	304	35.0	35.0	52.5
C-CMOS	28	06	448	896	272	544	368	39.3	39.3	58.9
HSPC	22	06	416	832	256	512	352	38.5	38.5	57.7
FA24T	24	06	384	768	240	480	336	37.5	37.5	56.3

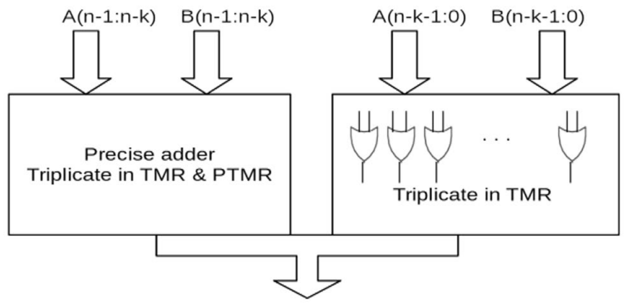


Fig. 4 Design idea for Fault tolerant Approximate adder

using TMR along with the LOA and in the second PTMR is used to augment the approximate adder. The design methodology for obtaining the fault tolerant approximate adders is given in Fig. 4. The entire adder comprising i.e., the precise and the imprecise sub-adder modules are included in a triplicated manner while designing the fault tolerant approximate adder based on TMR. In case of the PTMR based fault tolerant approximate adder, only the k-bits are triplicated while the remaining "n-k" bits are in the simplex mode.

The detailed implementation of the TMR based fault tolerant approximate adder is in Figs. 5 and 6 illustrates the implementation details of the PTMR based fault tolerant approximate adder. The building blocks or the basic cells in both these architectures are the Full adders (FA), Or gates and the voter circuit. A 1-bit voter circuit comprises of three (03) AND gates and one (01) OR gate.

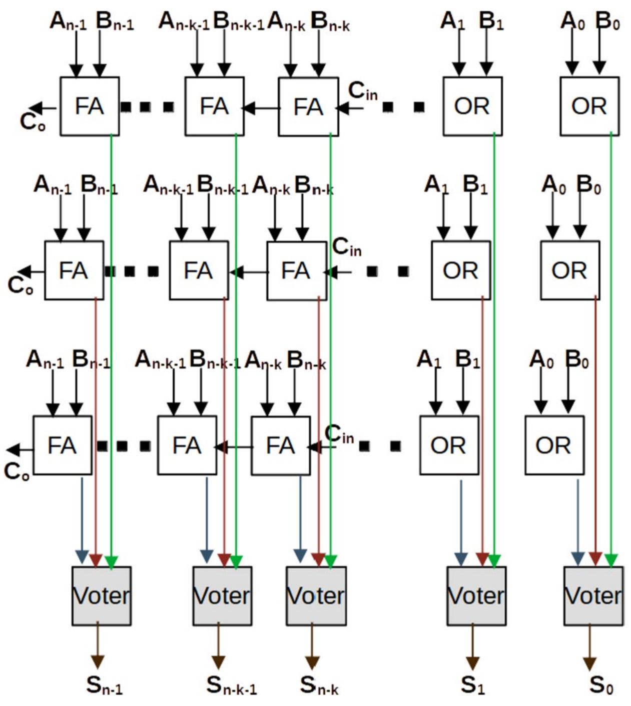


Fig. 5 TMR based fault tolerant approximate LOA

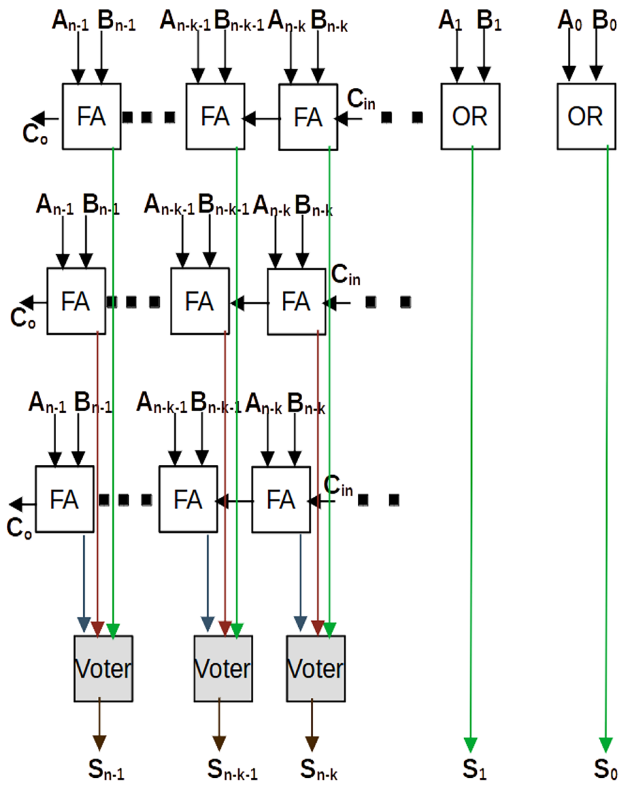


Fig. 6 PTMR based fault tolerant approximate LOA

The hardware requirements in a TMR based fault tolerant approximate adder is included in Table 4.

The PTMR based fault tolerant approximate adder is obtained by triplicating the hardware for the 'p' most significant bits while retaining the remaining hardware in a non-redundant manner. The value of 'p' in this quantitative analysis is taken as $p = 8$ and the hardware requirements calculated which are presented in Table 5.

The transistor requirements for the implementation of TMR & PTMR based fault tolerant approximate adders is quantitatively computed for the different adder structures. These calculations are done for $p = 8$ and $k = 8$ in PTMR for a 16-bit adder where p indicates the number of MSBs that are triplicated and k the width of the precise adder. For 32-bit adders the analysis is done for $k = 8$ and $k = 16$, with $p = 8$ in both the cases. The hardware requirement based on the transistor count for fault tolerant approximate adder

Table 4 Hardware requirements for TMR based fault tolerant approximate adder

Word size	**16-bit**	**32-bit**	
Basic cell	***k = 8***	***k = 8***	***k = 16***
1-bit full adder	24	48	24
OR gate	32	64	88

Table 5 Hardware requirements for PTMR based fault tolerant approximate adder

Word size →	16-bit	32-bit	
Basic cell ↓	***k* = 8**	***k* = 8**	***k* = 16**
1-bit full adder	24	32	24
OR gate	16	32	32
AND gate	24	48	24

is compared with the simple non-redundant fault intolerant Ripple Carry Adder (RCA). These are included in Table 6.

The table indicates that for certain adder structures (CPL, TGA, CCMOS, HSPC and FA24T) there is a reduction in the hardware (on the basis of transistor count) of a fault tolerant approximate adder when compared to a fault intolerant precise RCA.

4 Image Sharpening Using Fault Tolerant Approximate Adders

The efficacy of approximate computations in different applications has been demonstrated. The efficacy of the fault tolerant approximate adders for image sharpening applications and their improved performance in case of faults shall be established in this section.

4.1 Image Sharpening

The quality of the image visibly is influenced if the high-frequency elements are eliminated. If the high frequency components are instead enhanced, there is an improvement in the image quality. This concept has been used in improving the image quality in highlighting the edges of the image and its finer details. The algorithm is as given below:

If AI is the input image, then AK, the output image can be computed as:

$$AK\ (x,y) = 2AI(x,y) - K$$

where

$$K = \frac{1}{273}\sum_{p=-2}^{2}\sum_{q=-2}^{2} H(p+3,q+3)I(x-p,y-q)$$

and

$$H = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

The designed fault tolerant approximate adders are used in implementing the image sharpening algorithm [12]. This is implemented using MATLAB and the quality of the sharpened image is then measured. Peak Signal to Noise Ratio (PSNR) is an important parameter used to measure the quality of the image. It is based on the mean square error (MSE) of the reconstructed image which usually involves loss of information.

For an exact image, AI and an approximate image AK (where AI and AK are both monochrome images having m *by* n pixels) MSE is given as:

$$MSE = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[AI(i,j) - AK(i,j)]^2$$

The PSNR is then defined as:

$$PSNR = 20\ \log_{10}\frac{Max}{\sqrt{MSE}}$$

Table 6 Transistor count comparison for TMR & PTMR based fault tolerant approximate adders

Adder structures →			10T	14T	CPL	TFA	TGA	CCMOS	HSPC	FA24T
16-bit adders	***k* = 8**	TMR	384	480	912	528	624	816	672	720
		Add %	140	114	78	106	95	82	91	88
		PTMR	288	384	816	432	528	720	576	624
		Add %	80	71	59	69	65	61	64	63
32-bit adders	***k* = 8**	TMR	768	960	1824	1056	1248	1632	1344	1440
		Add %	140	114	78	106	95	82	91	88
		PTMR	416	544	1120	608	736	992	800	864
		Add %	30	21	09	15	15	11	14	13
	***k* = 16**	TMR	672	768	1200	816	912	1104	960	1008
		Add %	110	71	17	59	43	23	36	31
		PTMR	384	480	912	528	624	816	672	720
		Add %	20	07	-11	03	-03	-09	-05	-06

where MAX is the maximum value of the pixel in the image. For example, if it is encoded using 10-bits, the maximum value can be 1023.

4.2 Implementation Details

The adders are implemented and their performance evaluated on the basis of Structural Similarity Index Measure-SSIM, MSE and PSNR, in the absence and presence of faults using image processing. Image sharpening is used for enhancing the image. The human sensory system interpretation of the resultant images along with the values obtained for the above metrics are observed. These applications require many addition operations and hence are suitable for comparing the performance of fault intolerant and fault tolerant exact and approximate adders.

The experimental setup to perform the comparisons is shown in Fig. 7. It presents the steps involved in substantiating the reliability of the fault tolerant approximate adder. Initially the Image sharpening algorithm is implemented using MATLAB and then the test images are applied to the fault intolerant exact adder, approximate adder and the designed fault tolerant approximate adders. The output is checked in the absence of any fault and then after introducing a single stuck -at fault.

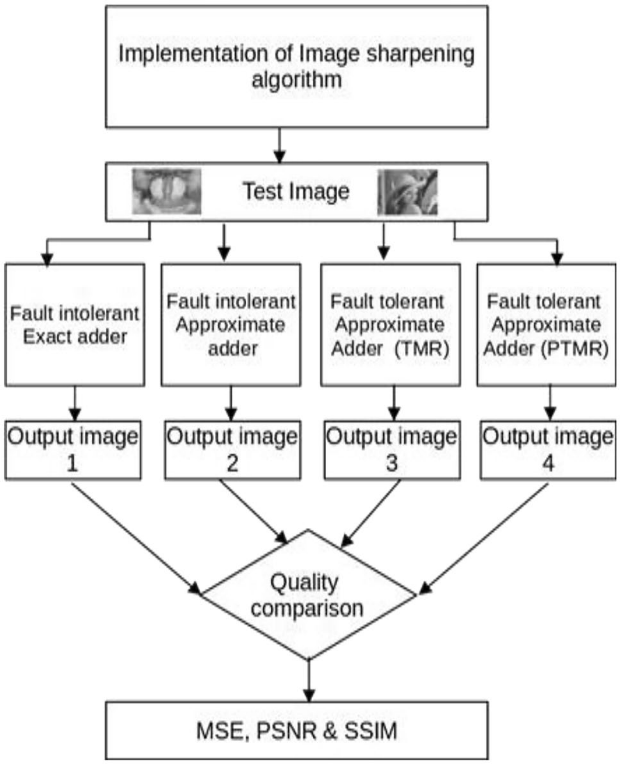

Implementation of Image sharpening algorithm
Test Image
Fault intolerant Exact adder
Fault intolerant Approximate adder
Fault tolerant Approximate Adder (TMR)
Fault tolerant Approximate Adder (PTMR)
Output image 1
Output image 2
Output image 3
Output image 4
Quality comparison
MSE, PSNR & SSIM

Fig. 7 Experimental set-up to evaluate fault intolerant and fault tolerant precise and approximate adders

The stuck-at fault model is used as it is able to model the majority of faults that may occur due to variations in transistor geometry, delayed switching, process variations etc. The operation is checked in the absence of fault and presence of single stuck-at-zero and single stuck-at-one faults in architecture 1 to obtain output image 1. The same is repeated by replacing the fault intolerant exact adder with the fault intolerant approximate adder to obtain output image 2, TMR based fault tolerant approximate adder to obtain output image 3 and PTMR based fault tolerant approximate adder to obtain output image 4. The images used in this experiment are the standard images – Baboon and Lena. In all these cases the SSIM, MSE & PSNR are observed and tabulated.

5 Results

The performance of the various adders is evaluated on the basis of Structural Similarity Index Measure (SSIM), Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) in the image sharpening algorithm. The image sharpening algorithm is applied to kernels of size 5×5 pixels, on the two images – Baboon and Lena. The visual quality of the images is first presented followed by the metric values. Figure 8 shows the original images.

The standard images of Lena and Baboon that are a usual standard are used to check the quality of the output. Initially the operation of the fault intolerant precise & imprecise adder along with fault tolerant imprecise adder is observed (Tables 7 and 8).

The output images obtained after image sharpening by using Baboon and Lena as the input image, in the presence of a single stuck-at fault (s-a-0 & s-a-1) are given in Figs. 9 and 10 respectively for architectures A1, A2, A3 and A4. These images help in checking the visual quality as perceived by the human eye. The results indicate that

Fig. 8 Original images of **a** Baboon and **b** Lena

Table 7 SSIM, MSE and PSNR values of the image Baboon for fault intolerant exact adder (A1), fault intolerant approximate adder (A2), TMR based fault tolerant approximate adder (A3) and PTMR based fault tolerant approximate adder (A4)

Image		**Baboon**							
Fault		***Architecture, A1***		***Architecture, A2***		***Architecture, A3***		***Architecture, A4***	
	Metric	*No Fault*	*Single Fault*	*No Fault*	*Single Fault*	*No Fault*	*Single fault*	*No Fault*	*Single fault*
S-a-0	***SSIM***	1	0.632498	0.993339	0.644741	0.993339	0.993339	0.993339	0.993339
	MSE	0	6.99E+03	1.20E+02	6.43E+03	1.20E+02	1.20E+02	1.20E+02	1.20E+02
	PSNR	Inf	9.686355	27.33258	10.04676	27.33258	27.33258	27.33258	27.33258
S-a-1	***SSIM***	1	0.001063	0.993339	9.58E-04	0.993339	0.993339	0.993339	0.993339
	MSE	0	1.91E+04	1.20E+02	1.91E+04	1.20E+02	1.20E+02	1.20E+02	1.20E+02
	PSNR	Inf	5.330808	27.33258	5.328538	27.33258	27.33258	27.33258	27.33259

Table 8 SSIM, MSE and PSNR values of the image Lena for fault intolerant exact adder (A1), fault intolerant approximate adder (A2), TMR based fault tolerant approximate adder (A3) and PTMR based fault tolerant approximate adder (A4)

Image		**Lena**							
Fault		***Architecture, A1***		***Architecture, A2***		***Architecture, A3***		***Architecture, A4***	
	Metrics	***Fault free***	***With fault***	***Fault free***	***With fault***	***Fault free***	***With fault***	***Fault free***	***With fault***
S-a-0	***SSIM***	1	0.704184	0.986957	0.711745	0.986957	0.986957	0.986957	0.986941
	MSE	0	5.49E+03	1.16E+02	4.55E+03	1.16E+02	1.16E+02	1.16E+02	1.21E+02
	PSNR	Inf	10.73166	27.49957	11.55325	27.49957	27.49957	27.49957	27.31647
S-a-1	***SSIM***	1	4.41E-04	0.986957	5.08E-04	0.986957	0.986957	0.986957	0.986957
	MSE	0	1.79E+04	1.16E+02	1.79E+04	1.16E+02	1.16E+02	1.16E+02	1.16E+02
	PSNR	Inf	5.590639	27.49957	5.592621	27.49957	27.49957	27.49957	27.49957

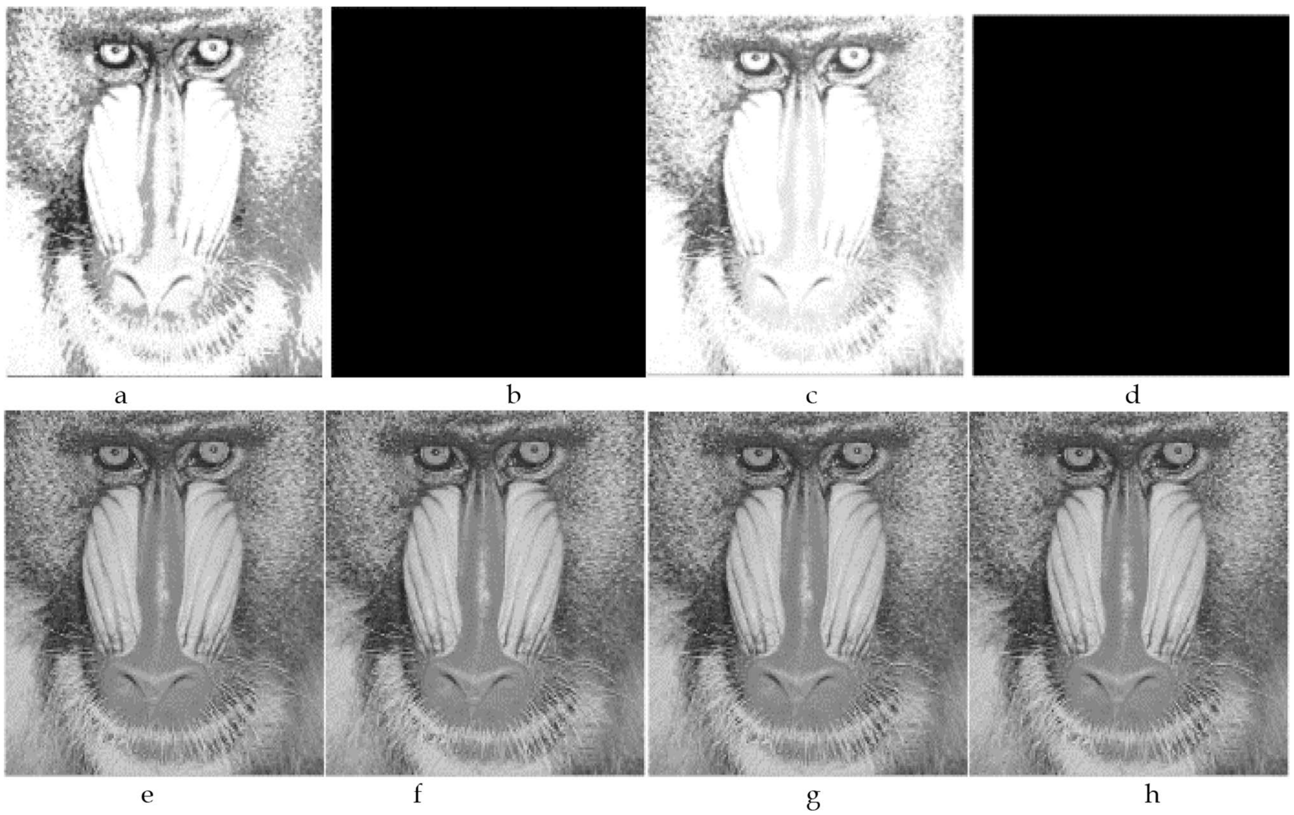


Fig. 9 Outputs with Lena as the input image. **a** Architecture 1 with single stuck-at-0 fault. **b** Architecture 1 with single stuck-at-1 fault. **c** Architecture 2 with single stuck-at-0 fault **d** Architecture 2 with single stuck-at-1 fault. **e** Architecture 3 with single stuck-at-0 fault. **f** Architecture 3 with single stuck-at-1 fault. **g** Architecture 4 with single stuck-at-0 fault. **h** Architecture 4 with single stuck-at-1 fault

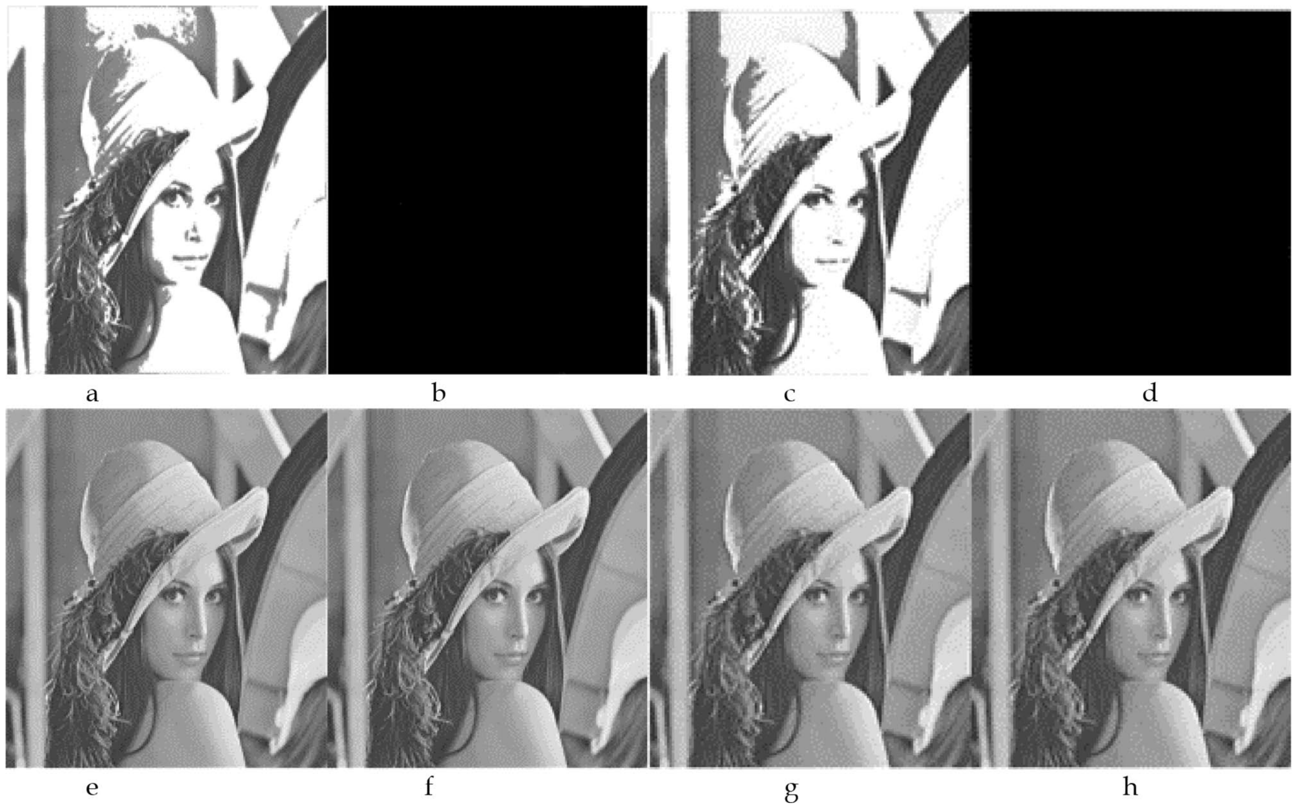


Fig. 10 Outputs with Lena as the input image. **a** Architecture 1 with single stuck-at-0 fault. **b** Architecture 1 with single stuck-at-1 fault. **c** Architecture 2 with a single stuck-at- 0 fault **d** Architecture 2 with single stuck-at-1 fault. **e** Architecture 3 with single stuck-at-0 fault. **f** Architecture 3 with single stuck-at-1 fault. **g** Architecture 4 with single stuck-at-0 fault. **h** Architecture 4 with single stuck-at-1 fault

the presence of faults has a detrimental effect on the visual quality of the image. In case of the fault intolerant precise adder and fault intolerant imprecise adder there is a visible degradation in the visual quality in case of a stuck-at-zero fault. This can be verified from Figs. 9a, c, and 10a, c. In case of a single stuck-at-one fault the entire image is indecipherable as can be ascertained from Figs. 9b, d, and 10b, d.

The visual quality perceived by the human eye does not indicate a significant difference among the four outputs. The PSNR, MSE & SSIM values for A3 and A4 which are fault tolerant imprecise adders, remains the same as that of a fault free adder even in the presence of a single fault. This is not the case for the fault intolerant adders, in which it can be observed that there is a drastic fall in these metrics in the presence of a fault.

6 Conclusion

The inclusion of fault tolerance in approximate adders can give a class of high-performance reliable adders. The TMR incorporates fault tolerance in both the precise and imprecise adder whereas the PTMR scheme protects the precise adder from failing due to faults. In practice the PTMR scheme would be more suitable for incorporating fault tolerance in approximate adders as the imprecise adder is expected to have inaccuracy.

The fault tolerant imprecise adders were used in an image processing application. The results obtained from an image sharpening algorithm implemented using the designed fault tolerant adders shows that there is a significant improvement in the quality of the outputs in case of a stuck-at fault, more so in the case of a stuck-at-one fault.

The QCA technology has potentially many advantages over CMOS but is prone to defects. The design of robust approximate adders may be implemented in QCA so that the advantages of this technology and the robustness of fault tolerant design can give a class of high-performance adders.

There are various optimizations possible by including different high speed or low power adders depending on the application. The utility of the fault tolerant adder is demonstrated using an image processing application. Its efficacy in different application that are error tolerant can be ascertained.

This seemingly simple but novel idea has the potential to significantly help in yield improvement while at the same time give reliable results without impacting the area, delay and power metrics. It is a paradigm that is significant for realizing reliable Approximate Computing units for critical image and video processing applications like telemedicine [11] and military applications [16] that is gaining prominence as digital transformation happens at a fast pace.

Funding No funding was received for conducting this study.

Data Availability All data generated or analyzed during this study are included in this published article.

Declarations

Ethical Approval Not applicable.

Conflicts of Interest The authors declare that they have no conflicts of interest.

References

1. Albicocco P, Cardarilli GC, Nannarelli A, Petricca M, Re M (2012) Imprecise arithmetic for low power image processing. In: 2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR). IEEE, pp 983–987
2. Alioto M, Cataldo GD, Palumbo G (2007) Mixed full adder topologies for high-performance low-power arithmetic circuits. Microelectron J 38(1):130–139
3. Chakradhar ST, Raghunathan A (2010) Best-effort computing: Re-thinking parallel software and hardware. In: Proceedings of the 47th Design Automation Conference. pp 865–870
4. Dokania V, Verma R, Guduri M, Islam A (2018) Design of 10T full adder cell for ultralow-power applications. Ain Shams Eng J 9(4):2363–2372
5. Ghosh S, Roy K (2010) Novel low overhead post-silicon self-correction technique for parallel prefix adders using selective redundancy and adaptive clocking. IEEE Trans Very Large Scale Integr (VLSI) Syst 19(8):1504–1507
6. Gupta V, Mohapatra D, Park SP, Raghunathan A, Roy K (2011) IMPACT: IMPrecise adders for low-power approximate computing. In: IEEE/ACM International Symposium on Low Power Electronics and Design. IEEE, pp 409–414
7. Harris D (2003) A taxonomy of Parallel Prefix networks. Thirty-seventh asilomar conference on signal, systems and computers. https://ieeexplore.ieee.org
8. Hasan M, Siddique AH, Mondol AH, Hossain M, Zaman HU, Islam S (2021) Comprehensive study of 1-bit full adder cells: review, performance comparison and scalability analysis. SN Appl Sci 3(6):644
9. Iqbal A, Manjunatha Chari K (2022) A brief analysis of fault-tolerant ripple carry adders with a design for reliable approximate adders. In: Proceedings of Second International Conference on Advances in Computer Engineering and Communication Systems: ICACECS 2021. Springer Nature Singapore, Singapore, pp 409–418
10. Javali RA, Nayak RJ, Mhetar AM, Lakkannavar MC (2014) Design of high speed carry save adder using carry lookahead adder. In: International Conference on Circuits, Communication, Control and Computing. IEEE, pp 33–36
11. Jridi M, Chapel T, Dorez V, Le Bougeant G, Le Botlan A (2018) SoC-based edge computing gateway in the context of the internet of multimedia things: experimental platform. J Low Power Electron Appl 8(1):1
12. Kumar UA, Sahith G, Chatterjee SK, Ahmed SE (2022) A high-speed and power-efficient approximate adder for image processing applications. J Circuits Syst Comput 31(03):2250049
13. Lin JF, Hwang YT, Sheu MH, Ho CC (2007) A novel high-speed and energy efficient 10-transistor full adder design. IEEE Trans Circuits Syst I Regul Pap 54(5):1050–1059
14. Liu C, Chu C, Xu D, Wang Y, Wang Q, Li H, Cheng KT (2021) HyCA: A hybrid computing architecture for fault-tolerant deep learning. IEEE Trans Comput Aided Des Integr Circuits Syst 41(10):3400–3413
15. Mahdiani HR, Ahmadi A, Fakhraie SM, Lucas C (2009) Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. IEEE Trans Circuits Syst I Regul Pap 57(4):850–862
16. Nomani T, Mohsin M, Pervaiz Z, Shafique M (2020) xUAVs: towards efficient approximate computing for UAVs—low power approximate adders with single LUT delay for FPGA-based aerial imaging optimization. IEEE Access 8:102982–102996
17. Parhi R, Kim CH, Parhi KK (2015) Fault-tolerant ripple-carry binary adder using partial triple modular redundancy (PTMR). In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp 41–44
18. Patel JH, Fung LY (1982) Concurrent error detection in ALU's by recomputing with shifted operands. IEEE Trans Computers 31(7):589–595
19. Radhakrishnan S, Nirmalraj T, Ashwin S, Elamaran V, Karn RK (2018) Fault tolerant carry save adders-A NMR configuration approach. In: 2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT). IEEE, pp 210–215
20. Raychaudhuri S (2008) Introduction to Monte Carlo simulation. In: 2008 Winter Simulation Conference. IEEE, pp 91–100
21. Reynolds DA, Metze G (1978) Fault detection capabilities of alternating logic. IEEE Trans Comput 12:1093–1098
22. Sanchez-Macian A, Martin-Toledano A, Bravo-Montes JA, Garcia-Herrero F, Maestro JA (2021) Reducing the impact of defects in Quantum-Dot Cellular Automata (QCA) approximate adders at nano scale. IEEE Trans Emerg Top Comput 10(2):635–647
23. Sasi G (2019) Fault tolerant design using 5-modular redundancy configuration with different voter circuits. CVR J Sci Technol 16(1):32–37
24. Shams AM, Darwish TK, Bayoumi MA (2002) Performance analysis of low-power 1-bit CMOS full adder cells. IEEE Trans Very Large Scale Integr (VLSI) Syst 10(1):20–29
25. Tsounis I, Agiakatsikas D, Psarakis M (2022) A methodology for fault-tolerant pareto-optimal approximate designs of fpga-based accelerators. ACM Trans Embed Comput Syst
26. Tung CK, Hung YC, Huang GS (2007) A low-power high-speed hybrid CMOS full adder for embedded system. In: 2007 IEEE Design and Diagnostics of Electronics Circuits and Systems
27. Vesterbacka M (1999) A 14-transistor CMOS full adder with full voltage swing nodes. In: 1999 IEEE Workshop on Signal Processing Systems
28. Von Neumann J (1956) Probabilistic logics and the synthesis of reliable organisms from unreliable components. Autom Studies 34(34):43–98
29. Weste NHE, Harris DM (2010) CMOS VLSI design: a circuits and systems perspective. Addison-Wesley, Boston, MA
30. Zimmerman R, Fichtner W (1997) Low-power logic styles: CMOS versus pass-transistor logic. IEEE J Solid State Circuits 32(7):1079–1090

31. Zhang M, Gu J, Chang CH (2003) A novel hybrid pass logic with static CMOS output drive full-adder cell. In: Proceedings of the 2003 International Symposium of Circuits and Systems
32. Zhu N, Goh WL, Zhang W, Yeo KS, Kong ZH (2009) Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. IEEE Trans Very Large Scale Integr (VLSI) Syst 18(8):1225–1229

Asma Iqbal completed her Bachelors in Electronics Engineering from Aligarh Muslim University, UP, India, Masters in Technology from JNTU, Hyderabad, India and has submitted her PhD thesis. She is also pursuing a 4-year online degree program BS (Data Science) from IIT Madras, India. She has 24 years of teaching engineering undergraduates and has guided many students in their project works. She has been a coordinator for national conferences organized in her department and published 10 papers.

Syed Affan Daimi is an undergrad student doing an offline BE (Computer Science) from Osmania University and online BS (Data Science) from IIT Madras, India. He has completed summer school at Oxford and interned for machine learning research project at IIT Hyderabad, India.

Dr. K. Manjunatha Chari did his B.E.(ECE) from Gulbarga University, M.Tech. (DSCE) from JNTU(H) and then his Doctorate from JNTU(K). He is currently Professor and earlier served as Head of ECE department at GITAM School of Technology, Hyderabad campus. He has more than 26 years of teaching and industrial experience. He has published numerous papers in national and international conferences and journals.